A Framework for Multimedia Communication in a General-Purpose Distributed System

David Anderson
Robert Wahbe

Report No. UCB/CSD 89/498
March 31, 1989

Computer Science Division (EECS)
University of California
Berkeley, CA 94720

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
31 MAR 1989		**00-00-1989 to 00-00-1989**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Framework for Multimedia Communication in a General-Purpose Distributed System	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Next-generation distributed computer systems will support network communication of digital multimedia (audio and video) traffic. Evolving hardware technology will provide the necessary performance. However, difficult design issues arise in the software structure of such a system. We consider three related issues: 1) What basic IPC abstractions can accommodate the likely range of traffic types? 2) How can IPC functions such as security and flow control be optimally assigned to system layers? 3) How can host resources such as CPU time and memory be optimally allocated? In response to these questions, we have defined channels, an IPC abstraction with security, reliability, and performance parameters. The channel abstraction underpins the network communication architecture of DASH, an experimental distributed system. In DASH, IPC functions are dynamically assigned to the layer in which they can be performed most efficiently, and the allocation policies for kernel resources are dictated by the performance needs of IPC. All IPC (local and request/reply as well as multimedia) is handled in a single framework.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**19**	

A FRAMEWORK FOR MULTIMEDIA COMMUNICATION IN A GENERAL-PURPOSE DISTRIBUTED SYSTEM

David P. Anderson
Robert Wahbe

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

March 31, 1989

ABSTRACT

Next-generation distributed computer systems will support network communication of digital multimedia (audio and video) traffic. Evolving hardware technology will provide the necessary performance. However, difficult design issues arise in the software structure of such a system. We consider three related issues: 1) What basic IPC abstractions can accommodate the likely range of traffic types? 2) How can IPC functions such as security and flow control be optimally assigned to system layers? 3) How can host resources such as CPU time and memory be optimally allocated?

In response to these questions, we have defined *channels*, an IPC abstraction with security, reliability, and performance parameters. The channel abstraction underpins the network communication architecture of DASH, an experimental distributed system. In DASH, IPC functions are dynamically assigned to the layer in which they can be performed most efficiently, and the allocation policies for kernel resources are dictated by the performance needs of IPC. All IPC (local and request/reply as well as multimedia) is handled in a single framework.

Sponsored by the California MICRO program, AT&T Bell Laboratories, Digital Equipment Corporation, IBM Corporation, Olivetti S.p.A, and the Defense Advanced Research Projects Agency (DoD) Arpa Order No. 4871. Monitored by Naval Electronic Systems Command under Contract No. N00039-84-C-0089.

1. INTRODUCTION

There is growing interest in incorporating multimedia[1] user interfaces and communication into general-purpose distributed operating systems [14, 24-27]. Such a system could support mixed-media documents and conferencing, and might replace telephones, answering machines, and CD players. Wide-area networks such as Broadband ISDN (BISDN) [8] may eventually allow these systems to be extended to a national or global scale. They could then subsume and integrate many existing communication media (mail, FAX, news and entertainment distribution) and support new applications such as interactive video.

DASH [2] is an experimental distributed system that combines multimedia communication with traditional distributed system functions such as program execution and RPC-based service access. The DASH design allows the integration of multimedia communication, computation, and storage. For example, digital audio streams can be directed among audio I/O devices, disk files and processes in much the same way that byte streams are used in UNIX.

The DASH operating system kernel provides a uniform abstract interface to networks, manages and multiplexes network communication, and uses software to enhance the capabilities of network hardware. In designing the communication facilities of DASH, we encountered several basic issues:

- What are the "right" IPC abstractions for multimedia and other traffic? A framework is needed in which 1) client needs can be expressed; 2) network capabilities can be expressed, and 3) these needs and capabilities can be compared meaningfully.
- How should the IPC system be layered, and how should functions such as error detection, security and flow control be assigned to the different layers?
- How can the operating system best allocate its local resources (CPU time, network queueing, and buffer space) to support real-time IPC? This issue is crucial because high-speed networks will probably shift performance bottlenecks into hosts.

To address these issues, we first developed an IPC abstraction called *channels*. The channel abstraction is used at several layers in DASH (see Figure 1a). Channels have security, reliability and performance parameters that are exchanged between layers. Using these parameters, the DASH kernel intelligently allocates local resources and assigns IPC functions to the layer that can do them most efficiently. Many software functions are consolidated in a *subtransport layer* that exports the characteristics of different network hardware, yet provides a uniform interface.

In contrast to DASH, the DARPA Internet model (shown in Figure 1b) provides an Internetwork Protocol (IP) layer with a datagram interface. The abstraction provided by IP is the lowest common denominator of the services provided by the component networks, and their other characteristics (such as security and performance) are hidden. Using the IP datagram service, transport protocols provide abstractions such as reliable byte streams and RPC [5, 18]. This architecture facilitates interconnection, but it does not allow any of the characteristics of the networks to be exported to higher layers. As a

[1] We use the term *multimedia* to refer primarily to digital audio and video.

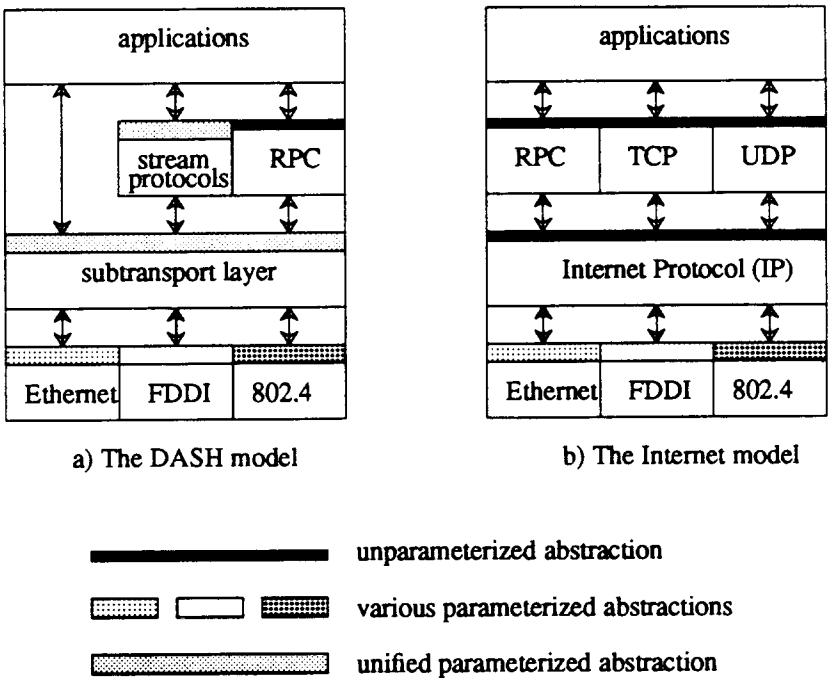

Figure 1: IPC abstractions in a) the DARPA Internet architecture and b) the DASH architecture.

result, it cannot support general real-time communication and it is inflexible with respect to placement of IPC functions.

The paper is organized as follows: Section 2 summarizes the performance requirements of multimedia traffic, surveys relevant hardware technology, and discusses the role of operating systems in multimedia communication. Sections 3 and 4 describe the channel abstraction and its use in the DASH communication architecture. The use of channel parameters in assigning IPC functions and allocating local resources is discussed in Sections 5 and 6, and Section 7 gives conclusions.

2. ARCHITECTURES, REQUIREMENTS AND RESOURCES

2.1. The Role of Operating Systems in Multimedia Communication

Many hardware/software architectures for multimedia communication are possible. At one extreme, voice and "data" traffic share a physical network, but have no other interaction (Figure 2a). The Xerox Etherphone system [25] provides a partially-integrated facility for voice communication and storage (Figure 2b). Connection establishment is controlled through a workstation-based user interface, but voice data is not handled by the workstation. The IMAL multimedia laboratory [14], uses an architecture that is similar except that a separate physical network is used. Analog video signals are

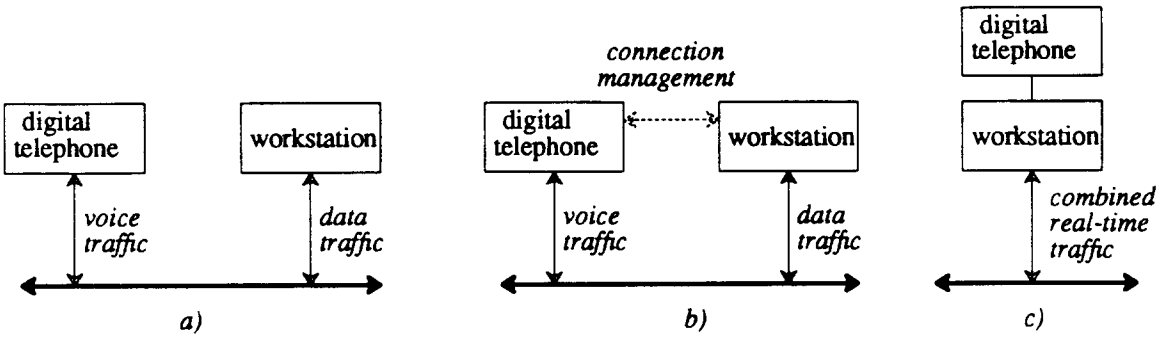

Figure 2: Hardware architectures for multimedia communication.

routed through a computer-controlled crossbar switch.

Integration and uniformity encourage synergy among system functions. In the UNIX operating system [19], for example, all I/O traffic (network, disk, terminal, and graphics output) passes through the main memory of the host, under the control of the operating system. Exploiting this uniformity, UNIX defines a standard programming interface and command syntax for communication and I/O, allowing processes to be conveniently connected to I/O devices or to other processes An analogous uniformity in multimedia communication (Figure 2c) would 1) increase the portability and generality of multimedia software, 2) support applications such as video hypertext [7] that use several media interactively, and 3) allow multiple physical networks (e.g., FDDI and BISDN) to be used transparently.

Is integrated multimedia communication technologically feasible? To address this question, we must first survey the performance needed by multimedia applications and the performance offered by evolving hardware technology.

2.2. Performance Requirements of Multimedia Applications

In its simplest form, multimedia traffic is sent uncompressed between I/O devices (converters) that have limited buffering. Data is sent in equal-sized units at equal time intervals. To avoid starvation and overrun, a channel with approximately constant delay is needed. The required throughput varies: 64 Kbps for voice-quality audio, 1.4 Mbps for CD-quality stereo audio, and anywhere from 1.6 Mbps to 1 Gbps or more for video, depending on the resolution and frame rate. The maximum acceptable delay depends on the application. In voice conversations a round-trip delay of more than 50 milliseconds causes undesirable echo effects [15], while a delay of a second would be acceptable for the transmission of a recorded program or message. The error rate, message loss rate, and reordering requirements may also vary with the application. If video is being transmitted for one-time viewing, a nonzero message loss rate (resulting in frames being shown twice in succession) may be acceptable. The error tolerance may be lower when

the transmission is for archival purposes.

Other types of multimedia traffic are *bursty* (i.e., have a variable data rate). The degree of burstiness may be expressed by deviation-to-average (DAR) or peak-to-average (PAR) ratios. Compression often introduces burstiness. For example, current compression techniques allow TV-quality video to be compressed to an average of 139 Kbps to 2.7 Mbps (using different coding schemes) with DAR of 2 to 3 [16]. The burstiness can be reduced by dynamically varying the picture quality. Silence suppression in typical voice conversations reduces the data rate by about 60% and introduces burstiness [6].

The requirements of applications that generate bursty traffic vary widely. If the receiver has limited buffering, the variance of the delay must be limited; in the extreme case an isochronous (constant delay) channel is needed. Otherwise it may suffice to have a bound on the delay, either in a statistical sense or as a hard deadline. For interactive video, delay is more important than packet loss, and peak delay is more important than average [16].

In addition to these performance and reliability requirements, multimedia traffic may have security requirements such as authentication and/or privacy. Finally, most "conventional" digital traffic (file access, remote login, window systems, etc.) has implicit real-time requirements as well. For example, a moderate amount of delay (10-100 ms) is acceptable for user interface traffic because of human perceptual limitations. The acceptable delay and throughput for file access depends on the client; virtual memory page-out will have much looser delay requirements than the page-in of an interactive program. In general, the requirements of conventional traffic are not as stringent as those of multimedia traffic, but the difference is quantitative rather than qualitative.

2.3. Evolving Hardware Technology

With the advent of fiber-optic transmission media and fast switching technology, next-generation networks will offer high data rates, and they may also offer real-time performance guarantees. As an example, Fiber Distributed Data Interface (FDDI) is a 100 Mbps fiber-based ring network capable of supporting 500 stations on a 100 Km ring [20]. In its enhanced (FDDI-II) mode, it provides up to 16 "isochronous channels" of 6.4 Mbps, "synchronous frames" with a small delay bound, and multiple priorities for other packets. On a larger scale, Broadband ISDN (BISDN) is an evolving international standard for fiber-based commercial networks [8, 11]. Standard channel bandwidths of 150 Mbps and 600 Mbps have been proposed. With both FDDI and ISDN, clients must "reserve" channels in advance.

Equally importantly, the performance of host I/O systems is increasing as well. Throughputs of 100 to 500 Mbps or more have been achieved or predicted for the major hardware components: memory busses [28], network interfaces [12], and permanent storage [17].

2.4. The Window of Scarcity

The general task of an operating system is to satisfy client needs using hardware resources. This is impossible if the needs exceed the resources (see Figure 3). If there are abundant resources, hardware is always fast (or capacious) enough to satisfy needs.

In the middle area, which we call the *window of scarcity*, the operating system must require clients to reserve resources, and carefully schedule resource usage, in order to satisfy typical client needs.

Based on the above discussion, we conjecture that the task of providing integrated multimedia communication is in the window of scarcity, and will remain there for at least the next 10-15 years. The demands of multimedia applications (and the scale of distributed systems) will increase at roughly the same rate as hardware capabilities. To satisfy client needs, if this conjecture holds, the hardware resources of the system (local and/or network) will have to be allocated and scheduled according to those needs.

3. CHANNELS: AN ABSTRACTION FOR REAL-TIME IPC

If a distributed system is to provide integrated multimedia communication, what IPC abstractions should be used? If the system is to function within the window of scarcity, the IPC abstraction must be *parameterized* to allow the expression of client needs and hardware capabilities. Abstractions built on datagrams (or virtual circuits) therefore cannot support multimedia communication in this case.

DASH uses an IPC abstraction called a *parameterized message channel*, or simply a *channel* [4]. A channel is simplex (unidirectional) entity[2] that carries messages (untyped byte arrays) from a sender to a receiver. The sender and receiver entities (processes or hosts) are called *clients*, and the system supporting the creation and use of channels is the

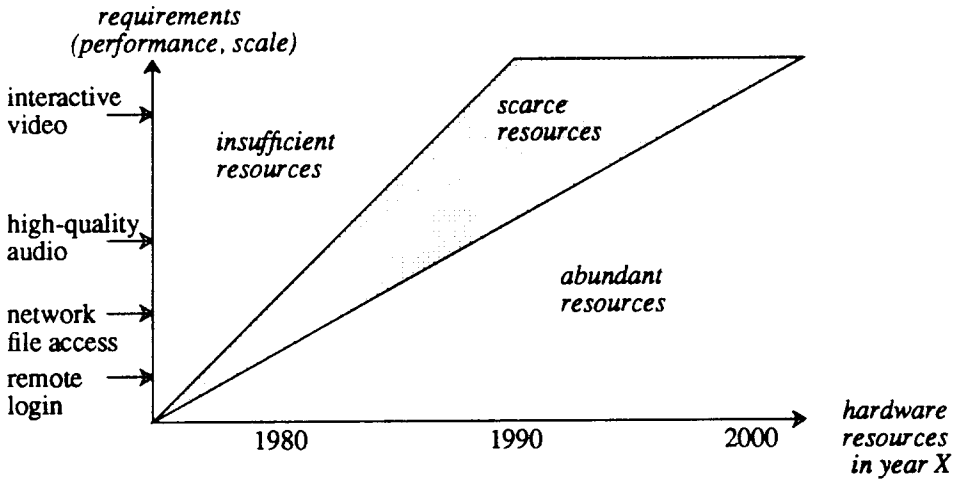

Figure 3: The window of scarcity.

[2] Channels are simplex because real-time traffic is often asymmetrical, as when audio data is downloaded from a data server. Asymmetrical channels (600 Mbps towards the "customer", 150 Mbps the other way) have been proposed for BISDN [11].

provider. A client of one level may be a provider to a higher level. A channel C has a set of parameters P_C describing its security, workload, performance, reliability and error characteristics. The parameters are fixed for the life of the channel.

To relate needs and resources, we need a basis for comparing parameter sets. For this purpose, a relation $\leq$ is defined on parameter sets. If $P_1 \leq P_2$ holds then P_2 is at least as good as P_1; i.e., P_2 is acceptable to a client whenever P_1 is acceptable. In cases where billing is done, P_1 must be no more expensive than P_2. $\leq$ is a partial order; two parameter sets may be incomparable under $\leq$.

A channel creation request includes parameter sets P_{min}, P_{target}, and P_{max}. P_{min} is the minimal needs of the client, and P_{max} is the maximum he is willing to pay for. The actual parameter set P of the resulting channel must satisfy $P_{min} \leq P \leq P_{max}$; the request is rejected if this is impossible. The P_{target} parameters are a "hint" to the channel provider, which tries to match them as closely as possible[3]. The creator of a channel (which may be either the sender or the receiver) is "billed" for the channel, if such a notion exists.

A channel *fails* when one of the clients' hosts crashes, or when a failure or resource scarcity in the network makes it impossible to continue providing the channel. When a failure occurs no further messages are delivered, and the surviving channel clients are notified of the failure.

3.1. Channel Parameters

A given parameter is either a *provider parameter* or a *client parameter*, depending on who is responsible for enforcing it. When a client violates a restriction, the provider might 1) notify the client; 2) temporarily suspend the offending send operation; 3) drop messages; 4) cancel some or all of the channel guarantees; 5) close the channel. The choice is itself a parameter, as is the choice of whether the provider is responsible for notifying the client if it fails to satisfy a guarantee.

What should be the set of channel parameters? Our criteria, roughly speaking, are that a property should be a parameter if: 1) some clients need it, but not all; 2) some networks provide it, but not all; 3) it is potentially expensive to provide it in software. We have chosen the following basic channel parameters:

parameter	*units*	*responsibility*
maximum message size	bytes	client
authentication	Boolean	provider
privacy	Boolean	provider
sequenced	Boolean	provider
acknowledgements	Boolean	provider
bit error rate	probability	provider

The $\leq$ relation involves these parameters in the obvious way. $P_1 \leq P_2$ only if the parameters of P_1 are at least as "strong" as those of P_2 (e.g., P_2 must sequenced if P_1 is).

[3] The design could be extended to allow multiple minimum and maximum parameter sets; the actual channel would be required to lie between *some* pair of minimum and maximum parameter sets.

There are many possible ways to parameterize the performance and workload of a communication channel. We chose a set of parameters that can express the needs of most types of multimedia traffic, and can express the facilities of some existing and proposed networks.

parameter	*units*	*responsibility*
delay bound	seconds	provider
delay mean bound	seconds	provider
delay variance bound	seconds	provider
average workload	bits/second	client
peak workload	bits/second	client
average loss rate	probability	provider
peak loss rate	probability	provider
channel class	(see below)	

The loss rates are the probability that a message is not delivered. (This loss may be due to buffer overrun or to checksum errors). The meaning of a channel's performance parameters depend on its *class*:

Guaranteed: the provider reserves all resources necessary to guarantee the performance parameters. A channel creation request is not granted if there are insufficient resources.

Best-available: like guaranteed, except the provider does not control all the resources used for the channel. As an example, hosts could provide a channel of this class over an Ethernet. The Ethernet media access protocol has no real-time guarantees [22], but the hosts can use the channel parameters to schedule the resources they do control: queueing priority and buffer space.

Best-effort: requests are always granted. The provider does not necessarily reserve resources (hosts and networks determine their own policies for dealing with starvation of best-effort channels). Delay bound parameters are used for scheduling (CPU, network queueing) based on message delivery deadlines (see Section 6).

A complete definition of $\leq$ is beyond the scope of this paper. The $\leq$ relation can only hold between classes in the following order:

$$\textit{best-effort} \leq \textit{best-available} \leq \textit{guaranteed}$$

Thus, a best-effort channel can never be substituted for a guaranteed channel, regardless of their parameters.

3.2. Examples of Channel Parameter Usage

Using the channel parameters listed above, one can express the multimedia performance requirements and resources discussed in Section 2 (for example, a periodic isochronous channel is sequenced, has zero delay variance, and has equal peak and average workloads). Using the best-available and best-effort classes, one can express the characteristics of channels in nondeterministic networks. The best-effort class can be used to express the needs of clients whose workload is unpredictable, that do not have rigid requirements, and for which resources should not be reserved.

Channel parameters are useful for non-multimedia traffic as well. For example, a request/reply protocol [23] can use channel parameters to assign appropriate message delay bounds (initial requests and replies have tighter bounds than retransmissions and acknowledgements) and to obtain appropriate retransmission intervals. It is not necessary to create a new channel for each operation; for example, the DASH RPC facility uses a set of long-lived ST channels. Stream protocols can potentially use channels to achieve greater performance or economy. A stream protocol for bulk data transfer [9] should use a high capacity, high delay channel for data. Acknowledgement and retransmission strategies can be simplified since round-trip delays may be estimated from channel parameters.

We will describe in later sections how an operating system can use channel parameters to intelligently assign IPC functions and schedule resources. In addition, the channel architecture may simplify the design of high-speed wide-area networks. Because workload information is available to the network, new approaches to routing, queueing, and congestion control may be possible [10].

4. THE DASH NETWORK COMMUNICATION ARCHITECTURE

The channel abstraction provides a basis for 1) optimal assignment of IPC functions to layers, and 2) optimal allocation of local resources. This is illustrated by the design of DASH, described in this section. The DASH network communication architecture [3] is based on channels. It consists of several layers (see Figure 4).

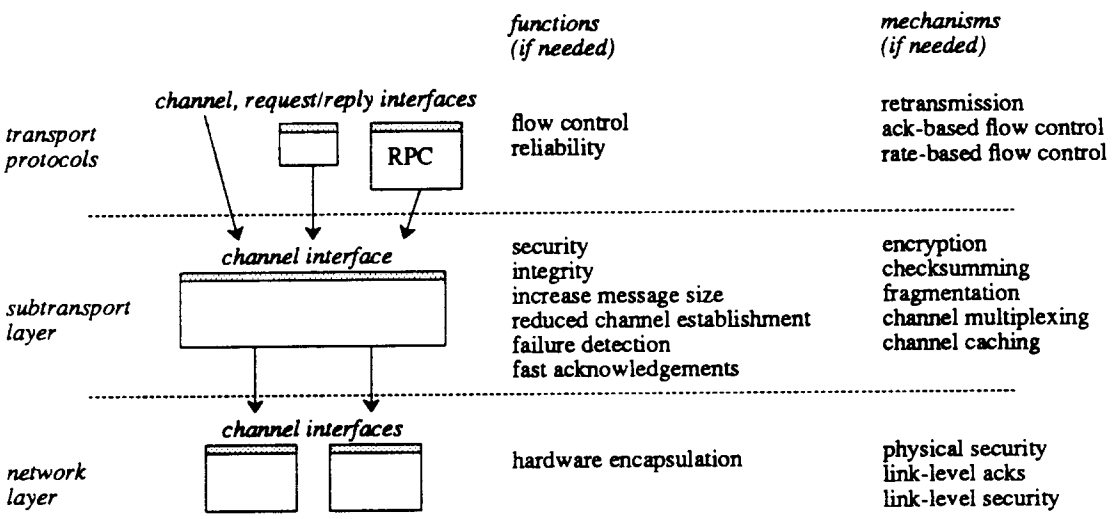

Figure 4: The DASH Communication Architecture.

The **network layer** encapsulates network-specific hardware and protocols. A DASH *network* is a logical entity (LAN, MAN, WAN, or internetwork) that provides channels. Each network to which a DASH host is connected is represented in its kernel by a software module exporting a standard interface offering host-to-host *network channels*. These modules encapsulate network-specific protocols for channel creation, deletion, and data transmission, as well as tasks such as routing and network management. They export flags (used by the subtransport layer to select security mechanisms) indicating whether hosts on the network are trusted, and whether the network is a physical broadcast network. A network supplies only those channel classes and parameters that it can support in hardware. For example, it provides security only if it is physically secure or does link-level encryption.

The **subtransport layer** (ST) manages, abstracts, and enhances the resources provided by the network layer. The ST provides *ST channels*. It uses software techniques to satisfy the client's error rate and security requirements, and to increase efficiency (see Section 4.1).

Clients (kernel and user processes) may directly create and access ST channels. If needed, protocols in the **transport layer** may be added to provide flow control and/or reliability. DASH RPC is a standard transport protocol. It provides a request/reply interface, and uses a set of ST channels whose parameters are chosen to optimize the protocol (see Section 3.2).

At higher levels, the DASH communication architecture includes 1) a *service access mechanism* providing a uniform interface to network services and other logical resources, and 2) a *global naming system* used to name long-lived entities such as hosts, services, and owners. The naming system is also a public key server (underpinning the ST's security mechanisms) and maps host names to network addresses.

The above layers involve network communication. Channels can also be used for communication between processes on a single host, in which case the role of the channel is to establish process scheduling deadlines.

4.1. The Functions of the Subtransport Layer

The ST layer plays three important roles in the DASH architecture. First, it provides a single standard interface to networks, even if a host is connected to multiple dissimilar networks. In this sense it is analogous to the IP layer of the Internet architecture.

Second, the ST acts as resource manager for network communication, matching client needs to network resources. The ST multiplexes ST channels onto network channels, and *caches* network channels (retaining a network channel even while it is not being used by any ST channel). This can reduce overhead and delay compared to a policy of creating a network channel for each ST channel, assuming that 1) during a short time period a host will tend to communicate repeatedly with a small set of remote hosts; 2) it is relatively costly to create network channels. The parameters of an ST channel may be different than those of the network channel on which it is multiplexed. The ST uses the $\leq$ relation in making multiplexing decisions; other policies (such as how much excess capacity to allocate when a network channel is created, and how long to maintain idle net channels) are host-dependent.

Third, the ST uses software techniques to "enhance" network channels, bridging the gap between client needs and network facilities. The channel-enhancement functions include the following (see [3] for a complete list):

Security and Reliability: If an ST channel has stricter security or reliability requirements than its network channel, the ST uses the most efficient method (encryption, cryptographic checksumming, checksumming, or encrypted trailer) to bridge the gap. These methods provide host-to-host security. The ST also allows clients to authenticate individual security principals. It maintains a "cache" of principals who have been authenticated to or from each host with which it has communicated.

Increased Maximum Message Size: The ST does fragmentation and reassembly, reducing the number of high-level messages and hence protocol processing and scheduling overhead. The maximum message size of an ST channel may be larger than that of its network channel, but not so large that the ST is forced to use retransmission or flow-control mechanisms to satisfy other requirements.

Client process parallelism and multiplexing: The ST's CPU-intensive tasks (checksumming and encryption) are done in parallel on multiprocessors. The ST manages this parallelism, using either client processes or its own "worker" processes. The ST also allows the sender to specify a process to awaken on message receipt (this is used for RPC reply messages, which are carried on a single ST channel but are directed to one of many client process).

In typical systems, the functions of ST are done at the transport or network levels. By centralizing the functions in the ST, the DASH architecture allows some of them to be done more efficiently. For example, a single secure channel between hosts can be maintained more efficiently than one per operation or session [1]. Furthermore, the kernel is smaller and simpler because functions are consolidated in a single module, rather than implemented separately in multiple transport or network modules.

Mechanisms for reliability and flow control are omitted from the ST because there are many tradeoffs and design decisions in these mechanisms, and we did not want to hardwire them into the ST.

5. ASSIGNMENT OF IPC FUNCTIONS

The DASH architecture gives the system two important degrees of freedom in supporting communication. First, the ST can choose the best network to use for each channel. For example, suppose a DASH host is connected to a MAN that is part of an internetwork and supplies its own native protocols as well. The native protocols provides a greater range of channel parameters than those of the internetwork. The host would view these as separate networks, represented by a MAN module and an internetwork module. The ST would use the MAN in preference to the internetwork for channels to hosts on the MAN.

Second, the DASH architecture allows communication functions such as flow control, error recovery, and security to be handled optimally, according to the following principles:

- If the function is not needed, it is not performed at all.

- If the function is needed, it is assigned to the layer that can do it most efficiently.

For example, suppose a client requires data privacy. It requests an ST channel with the *private* attribute. ST creates a new ST channel, multiplexing it onto a network channel. Several scenarios are possible:

(1) If the network is physically secure, no data encryption is done at any level.

(2) If the network has link-level encryption hardware, ST learns this from the network channel parameters, and does no data encryption.

(3) Otherwise privacy is provided by the ST using data encryption.

In any case, the optimal mechanism is used. If a client does not require privacy, no mechanism is used (which is again optimal).

Next, consider data integrity (bit error rate). In some architectures this is provided by checksumming in transport protocols. However, multimedia applications may tolerate errors, and in any case fiber optic networks may offer bit error rates so low that checksumming is not needed [21]. If encryption is being done at some level (see above) checksumming is not needed since encryption also provide error detection. Again, the DASH architecture allows the optimal choice to be made.

Finally, consider flow control (i.e., workload enforcement). This is the responsibility of the channel client. Depending on the channel parameters and the speeds of the data source and sink, a flow control mechanism may be unnecessary. If so, transport protocols can be simplified and made more efficient (or eliminated altogether).

6. LOCAL RESOURCE ALLOCATION

End-to-end network IPC involves resources local to the sending and receiving hosts (buffer space, CPU time, I/O bus bandwidth, and so on). With increasing network speeds, these resources will often be the performance bottleneck, so their allocation determines the performance available to users. In this section we sketch the resource allocation policies of the DASH kernel.

When a channel is created in DASH, its mean delay is divided among its various stages. (In the case of an ST channel, these stages include ST send processing, network queueing, network transmission, ST receive processing, and receiver process scheduling delay.) When a message is sent on a channel, there is a *deadline* by which it must be handled by the next stage (i.e., processed by a protocol, sent on a lower-level channel, or transmitted on a network medium). To a first approximation, this deadline is the current real time plus the delay allocated to the next stage[4]. These deadlines are used to prioritize resource usage (i.e., the order in which queued messages are transmitted, and the order in which processes are executed).

This strategy is an extension of deadline-based process scheduling [13], and shares its property of meeting deadlines if possible. However, the existence of multiple channel classes adds complexity. Guaranteed and best-effort messages may have intermixed deadlines, but best-effort traffic must not interfere with guaranteed traffic. DASH addresses this problem by using channel class to prioritize traffic, and using deadlines to

[4] The deadline could also reflect the amount by which the message is ahead of, or behind, schedule.

prioritize traffic within a class. For example, the DASH network module for Ethernet supports best-available and best-effort channels. The module maintains separate queues for outgoing packets of each class. Since the size and deadline of each packet are known, a "maximum start time" can be computed. A best-effort packet is transmitted only if it will finish before the earliest deterministic packet maximum start time (see Figure 5).

The DASH process scheduler also uses a hybrid priority/deadline mechanism. Processes involved in IPC are labeled by their channel class, and are assigned deadlines based on channel parameters. The scheduler maintains two deadline-sorted queues of runnable processes, one for best-effort and one for guaranteed and best-available. The scheduling policy is to run the process with the earliest deadline, except if the current real time exceeds the maximum start time of a higher-priority process. In some cases it is necessary to reserve a fraction of the sender and/or receiver host's total CPU time for a channel when it is created.

To demonstrate the effectiveness of these policies, we set up an experiment using the DASH kernel running on Sun 3/50 workstations connected by a dedicated 10 Mbps Ethernet. A single guaranteed channel with a delay bound of 10 ms and a periodic workload (a 1 Kbyte message every 10 ms) competes with random (Poisson) traffic on a best-effort channel. This is representative of a combination of high-quality audio channel and other traffic (file access, for example). As shown in Figure 6, the delay of the guaranteed channel remains below 10 ms, while the delay of the best-effort channel becomes infinite when its workload exceeds the remaining available throughput. The bottleneck resource in this case was CPU time. With faster CPUs, the network queueing policy would be exercised instead.

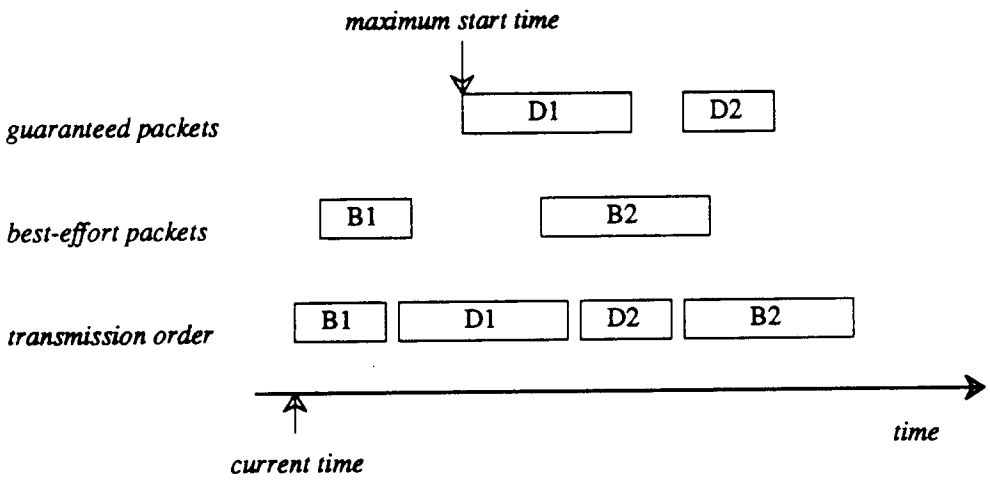

Figure 5: The queueing policy for guaranteed and best-effort packets. D2 is transmitted before B2 even though its deadline is later.

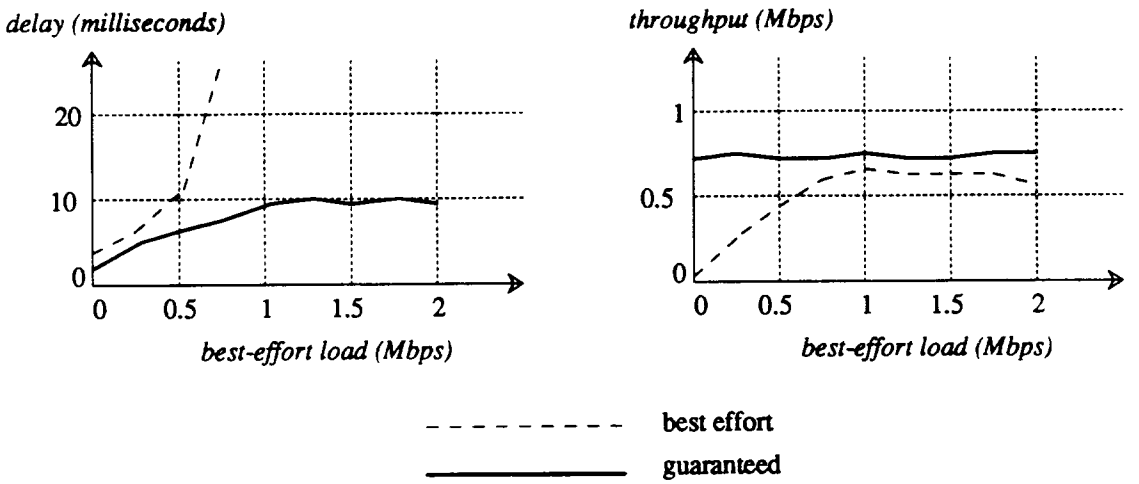

Figure 6: A demonstration of the DASH queueing mechanisms.

7. CONCLUSION

Successive generations of operating systems have been characterized by their IPC paradigms: byte streams, RPC, object invocation, and so on. The next generation of distributed operating system must support multimedia IPC, allowing application programs to direct streams of digital audio and video data between I/O devices, processes, permanent storage, and remote hosts. We have proposed *channels* as a standard IPC abstraction for such systems. Channels allow the client and the provider to negotiate performance, workload, reliability and security parameters. The abstraction provides a flexible parameterization, so that new network types, new I/O technologies, and new applications can be integrated easily.

We have also described the channel-based DASH communication architecture. Channel parameters in three ways in the DASH architecture: 1) they dictate how user-requested channels (ST channels) can be multiplexed onto network channels. 2) they dictate whether IPC functions (security, data integrity, flow control) are needed, and if so they determine the optimal layer in which to perform the function; 3) they allow local resources such as network bandwidth and CPU time to be scheduled intelligently.

The design described in this paper is still a rough framework. Many issues remain: 1) different performance parameters may be needed; 2) the restriction that parameters are fixed during the life of the channel may not be reasonable in the presence of network congestion and changing buffer sizes; 3) we have neglected bounds on setup time of channels and on the expected lifetime of channels; 4) multicast is absent from the design, and it is likely that some multimedia applications (e.g., broadcasts of TV programs) will require multicast services at a low level; 5) the design of programmer and user interfaces based on channels remains to be investigated.

REFERENCES

1. D. P. Anderson and P. V. Rangan, "A Basis for Secure Communication in Large Distributed Systems", *IEEE Symposium on Security and Privacy*, Apr. 1987.
2. D. P. Anderson and D. Ferrari, "The DASH Project: An Overview", Technical Report No. UCB/CSD 88/405, Computer Science Div., EECS Dept., Univ. of California at Berkeley, Feb. 1988.
3. D. P. Anderson and R. Wahbe, "The DASH Network Communication Architecture", Technical Report No. UCB/CSD 88/462, Computer Science Div., EECS Dept., Univ. of California at Berkeley, Nov. 1988.
4. D. P. Anderson, "A Software Architecture for Network Communication", *Proc. of the 8th International Conference on Distributed Computing Systems*, San Jose, California, June 1988.
5. A. Birrell and B. Nelson, "Implementing Remote Procedure Calls", *ACM Trans. Computer Systems 2*, 1 (Feb. 1984), 39-59.
6. P. T. Brady, "A Statistical Analysis of On-Off Patterns in Sixteen Conversations", *Bell System Technical Journal 47* (Jan. 1968).
7. S. Brand, *The Media Lab*, Viking, New York, 1987.
8. W. R. Byrne, T. A. Kilm, B. L. Nelson and M. D. Soneru, "Broadband ISDN Technology and Architecture", *IEEE Network*, Jan. 1989, 23-28.
9. D. D. Clark, M. L. Lambert and L. Zhang, "NETBLT: A High Throughput Transport Protocol", *Proc. of ACM SIGCOMM 87*, Stowe, Vermont, Aug. 1987, 353-359.
10. D. Ferrari, "Guaranteeing Performance for Real-Time Communications in Wide-Area Networks", Technical Report No. UCB/CSD 89/485, Jan. 1989.
11. R. Handel, "Evolution of ISDN Towards Broadband ISDN", *IEEE Network*, Jan. 1989, 7-13.
12. H. Kanakia and D. R. Cheriton, "The VMP Network Adapter Board (NAB): High-Performance Network Communication for Multiprocessors", *ACM SIGCOMM 88*, Aug. 1988, 175-187.
13. C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", *J. ACM 20*, 1 (1973), 47-61.
14. L. F. Ludwig and D. F. Dunn, "Laboratory for Emulation and Study of Integrated and Coordinated Media Communication", *Proc. of ACM SIGCOMM 87*, Stowe, Vermont, Aug. 1987, 283-291.
15. R. H. Moffett, "Echo and Delay Problems in Some Digital Communication Systems", *IEEE Communications Magazine 25*, 8 , 41-47.
16. N. Ohta, M. Nomura and T. Fujii, "Variable Rate Video Encoding Using Motion-Compensated DCT for Asynchronous Transfer Mode Networks", *IEEE International Conference on Communications*, 1988.
17. J. Ousterhout and F. Douglis, "Beating the I/O Bottleneck: A Case for Log-Structured File Systems", *Operating System Review 23*, 1 (Jan. 1989), 11-28.

18. J. Postel, "Transmission Control Protocol", *DARPA Internet RFC 793*, Sep. 1981.

19. D. M. Ritchie and K. Thompson, "The UNIX Time-Sharing System", *Comm. of the ACM 17*, 7 (July 1974), 365-375.

20. F. E. Ross, "FDDI - A Tutorial", *IEEE Communications Magazine*, May 1986, 10-15.

21. S. Shimada, K. Nakagawa and I. Takeshi, "Gigabit/s Optical Fiber Transmission Systems - Today and Tomorrow", *IEEE Int. Conf. on Comm.*, June 1986, 1538-1542.

22. J. F. Shoch and J. A. Hupp, "Measured Performance of an Ethernet Local Network", *Comm. of the ACM 23*, 12 (Dec. 1980), 711-721.

23. A. Spector, "Implementing Remote Operations Efficiently on a Local Network", *Comm. of the ACM 25*, 4 (Apr. 1982), 246-260.

24. T. Suzuki, H. Taniguchi and H. Takada, "A Real-Time Electronic Conferencing System Based on Distributed UNIX", *Proceedings of the 1986 Summer USENIX Conference*, Atlanta, Georgia, June 9-13, 1986, 189-199.

25. D. B. Terry and D. C. Swinehart, "Managing Stored Voice in the Etherphone System", *ACM Trans. Computer Systems 6*, 1 (Feb. 1988), 3-27.